Exhibit I

MATTHEW D. POWERS (Bar No. 104795)
matthew.powers@weil.com
EDWARD R. REINES (Bar No. 135960)
edward.reines@weil.com
JEFFREY G. HOMRIG (Bar No. 215890)
jeffrey.homrig@weil.com
JILL J. HO (Bar No. 236349)
jill.ho@weil.com
WEIL, GOTSHAL & MANGES LLP
Silicon Valley Office
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

ELIZABETH WEISWASSER (admitted pro hac vice)
elizabeth.weiswasser@weil.com
WEIL, GOTSHAL & MANGES LLP
767 Fifth Avenue
New York, NY 10153
Telephone: (212) 310-8000
Facsimile: (212) 310-8007

Attorneys for Plaintiff
NETWORK APPLIANCE, INC.,

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

NETWORK APPLIANCE, INC. Plaintiff, v. SUN MICROSYSTEMS, INC. Defendant.	Case No. 3:07-CV-06053-EDL **NETWORK APPLIANCE'S PATENT LOCAL RULE 3-1 DISCLOSURE OF ASSERTED CLAIMS AND PRELIMINARY INFRINGEMENT CONTENTIONS FOR U.S. PATENT NO. 7,200,715**

In accordance with N.D. California Patent Local Rule 3-1, Network Appliance, Inc. ("NetApp") hereby makes the following Disclosure of Asserted Claims and Preliminary Infringement Contentions for U.S. Patent No. 7,200,715 ("the '715 Patent"). These disclosures are made based on information ascertained to date, and NetApp reserves the right to modify or amend the disclosures contained herein based on the Court's claim construction or to reflect information that is ascertained in the future.

A. Patent Local Rule 3-1(a) Disclosures

Based upon presently known information, NetApp contends that at least the following claims of the '715 Patent have been infringed by Sun Microsystems, Inc. ("Sun") (collectively, "the Asserted Claims"): 21-22, 24-26, 33-34, 39, 41-45, 49-50, and 52.

B. Patent Local Rule 3-1(b) Disclosures

Based on information presently known, NetApp asserts infringement of every claim identified in response to Patent Local Rule 3-1(a) above by Sun's ZFS, as well any Sun's systems, devices or server platforms that incorporate, utilize, or execute ZFS, including, but not limited to, Sun Blade series of products, Sun Enterprise series of products, Sun Fire series of products, Sun Netra series of products, and Sun Ultra series of products ("Sun's Products").

NetApp reserves the right to update its infringement contentions to reflect the results of discovery in this matter.

C. Patent Local Rule 3-1(c) Disclosures

Please see "Appendix A," attached hereto and incorporated by reference, for a chart that identifies where each element of each of the Asserted Claims of the '715 Patent is found within Sun's Products.

D. Patent Local Rule 3-1(d) Disclosures

Based on presently known information, NetApp contends that Sun's Products identified in response to Patent Local Rule 3-1(b) all literally infringe the Asserted Claims. Additionally, NetApp contends that the products identified in response to Patent Local Rule 3-1(b) all infringe the Asserted Claims under the doctrine of equivalents.

E. Patent Local Rule 3-1(e) Disclosures

1. All of the Asserted Claims of the '715 Patent are entitled to a priority date of March 21, 2002, based on U.S. Application No. 10/105,034.

F. Patent Local Rule 3-1(f) Disclosures

NetApp asserts that Data ONTAP versions 6.2 and higher and NetApp platforms running Data ONTAP versions 6.2 and higher embody the claimed subject matter of each asserted claim of the '715 Patent.

DATED: January 22, 2008.

WEIL, GOTSHAL & MANGES LLP

/s/ Edward R. Reines___ ___________
Edward R. Reines
Attorneys for NetApp
NETWORK APPLIANCE, INC.

APPENDIX A

INFRINGEMENT CONTENTIONS
FOR U.S. PATENT NO. 7,200,715

Claim Language	Infringing Structure
21. A method for controlling storage of data, comprising:	ZFS implements a method for controlling the storage of data, as set forth below.
receiving one or more write requests associated with data blocks;	ZFS is a file system that receives a write request associated with data blocks.
receiving topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system;	The ZFS routine vdev_raidz_map_alloc() receives a pointer to the zio_t data structure, which includes topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system.
associating the data blocks with one or more storage blocks across the plurality of stripes as an association; and	The vdev_raidz_map_alloc() routine generates a raidz_map_t data structure, which associates data blocks to be stored to storage blocks located in multiple stripes.
writing the data blocks, in response to the association, to the one or more storage devices in a single write request.	The vdev_raidz_io_start() routine uses the raidz_map_t data structure to write the data blocks to the storage devices, as part of a single write transaction (i.e., a single write request).
22. The method of claim 21, further comprising: transmitting the association to a storage device manager.	The vdev_raidz_io_start() routine is passed a pointer to the raidz_map_t data structure (i.e., association).
24. The method of claim 21, further comprising: storing the data blocks in the association.	The vdev_raidz_io_start() routine stores a reference to the data blocks in the raidz_map_t data structure (i.e., association)
25. The method of claim 21, further comprising: storing the data blocks in a memory of the storage system.	ZFS stores data blocks in a memory of the storage system.
26. The method of claim 21, further comprising: creating an array as the association.	The raidz_map_t structure (i.e., association) is an array.
33. The method of claim 21, further comprising: using a plurality of disks for the storage system.	ZFS uses a plurality of disks as the one or more storage devices.
34. The method of claim 21, further comprising: transmitting the association to	The vdev_raidz_io_start() routine is passed a pointer to the raidz_map_t data structure.

Claim Language	Infringing Structure
the storage device manager.	
39. A storage system, comprising: a file system, the file system to receive one or more write requests associated with data blocks;	A storage system utilizing ZFS comprises a file system for receiving one or more write requests associated with blocks, as set forth below.
a storage device manager,	The storage device manager is ZFS.
the storage device manager to generate topological information of storage blocks configured in a plurality of parallel stripes of one or more storage devices, and	ZFS generates the zio_t data structure, which includes topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system.
to send the topological information to the file system;	ZFS is also the file system. The ZFS routine vdev_raidz_map_alloc() receives a pointer to the zio_t data structure.
and an association generated in the file system,	ZFS comprises a raidz_map_t data structure (i.e., association).
the association to associate the data blocks with one or more storage blocks across the plurality of stripes of the one or more storage devices,	The raidz_map_t structure maps the data blocks in a write request to storage blocks located in multiple stripes.
the association to be sent to the storage device manager,	Another ZFS routine, vdev_raidz_io_start(), is passed a pointer to the raidz_map_t data structure.
the storage device manager to write the data blocks, in response to the association, to the one or more storage blocks as a single write request.	The vdev_raidz_io_start() routine uses the raidz_map_t data structure to write the data blocks to the storage devices, as part of a single write transaction (i.e., a single write request).
41. The storage system of claim 39, further comprising: a memory to buffer the data blocks for the write request to the one or more storage devices.	A storage system utilizing ZFS comprises a memory to buffer the data blocks for the write request to the one or more storage devices.
42. The storage system of claim 39, further comprising: a memory to store the association containing the data blocks.	A storage system utilizing ZFS comprises a memory to store the raidz_map_t structure (i.e., association).
43. The storage system of claim 39, further comprising: one or more storage devices having storage blocks	A storage system utilizing ZFS comprises one or more storage devices having storage blocks.
44. The storage system of claim 39, further comprising: an array as the association.	The raidz_map_t structure (i.e., association) is an array.
45. The storage system of claim 39, further comprising: a buffer in the file system to	A storage system utilizing ZFS comprises a buffer in the file system to receive a write

Claim Language	Infringing Structure
receive the one or more write requests.	request.
49. The storage system of claim 39, further comprising: a plurality of disks as the one or more storage devices.	A storage system utilizing ZFS comprises a plurality of disks as the one or more storage devices.
50. The storage system of claim 39, further comprising: a RAID system as the plurality of storage devices.	A storage system utilizing ZFS comprises a RAID system as the plurality of storage devices.
52. A computer readable media, comprising: the computer readable media containing instructions for execution in a processor for the practice of the method of,	ZFS is implemented by means of program instructions that are stored on a computer readable medium, and that are executed by a processor for the practice of the method, as set forth below.
receiving one or more write requests associated with data blocks;	ZFS receives a write request associated with data blocks.
receiving topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system;	The ZFS routine vdev_raidz_map_alloc() receives a pointer to the zio_t data structure, which includes topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system.
associating the data blocks with one or more storage blocks across the plurality of stripes as an association;	The vdev_raidz_map_alloc() routine generates a raidz_map_t data structure, which associates data blocks to be stored, to storage blocks located in multiple stripes.
and writing the data blocks, in response to the association, to the one or more storage devices in a single write request.	The vdev_raidz_io_start() routine uses the raidz_map_t data structure to write the data blocks to the storage devices, as part of a single write transaction (i.e., a single write request).